

EXHIBIT P

JOHN WELCHES

3030 SATURN STREET • SUITE 202 • BREA • CALIFORNIA • 92821
PHONE (657) 258-0015 • E-MAIL JOHN.WELCHES@REDMALLARD.COM

OBJECTIVE

Red Mallard, Inc. was created to provide expert consulting and services in the following areas: Branding, Digital Experiences, Web Development, Print Management, and Content Strategy and Marketing.

EDUCATION

2003 - 2005 University of California, Riverside • Riverside, CA

Bachelor of Arts, Creative Writing

- Graduated w/ Honors
- Winner Abraham L. Polonsky Memorial Award
- Co/Winner Mosaic Award in Fiction

1999 - 2003 Fullerton College • Fullerton, CA

AA Liberal Arts/Business Studies

WORK EXPERIENCE

2014 - Present Red Mallard, Inc. Brea, CA

Founder & President

- Lead staff and team of consultants/freelancers to create excellent marketing and technology solutions.

2008 - 2014 J. Welches and Associates Orange, CA

Principal

- Created branding and marketing materials for organizations.
- Worked briefly for two companies during this time as a full-time marketer and editor: TKO Art (Newport Beach) and Walter Foster Publishing (Lake Forest).

2005 - 2008 Hobbs/Herder Advertising Newport Beach, CA

Writer/Account Executive • Web Services Consultant

- Initially worked as a creative team lead creating over 100 personal brands for professionals across North America. Promoted to assist clients in achieving their web marketing goals.

SKILLS

Writing Services • Leadership • Editing • Brand Messaging

REFERENCES

References and letters of recommendation available on request.

PERRY FRASER LLP

945 Mayo Street

Los Angeles, California 90042

Phone: (213) 324-4206

E-Mail: neil53@mac.com; jperry@2tg.co.uk

Attorney for Plaintiff

UNITED STATES DISTRICT COURT

NORTHERN DISTRICT OF CALIFORNIA

NATTO IYELA GBARABE, for Himself and
For Others Similarly Situated,

Plaintiff,

vs.

CHEVRON CORPORATION, a California
Corporation,

Defendant.

CASE NO.: 14-cv-00173-SI
(Assigned to Hon. Susan Illston – Courtroom
One)

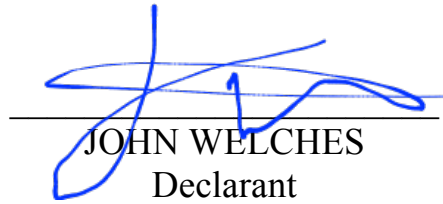
**DECLARATION OF JOHN WELCHES IN
SUPPORT OF REPLY TO
DEFENDANTS'S OPPOSITION TO
MOTION FOR CLASS CERTIFICATION;
DEMURRER TO PLAINTIFFS'
ORIGINAL COMPLAINT.**

I, JOHN WELCHES, declare and state as follows:

1. I am President of Red Mallard, Inc., a California corporation providing digital media and marketing solutions.
2. I have been retained by Perry Fraser LLP to coordinate and produce a digital platform which incorporates various security measures designed to accurately collect survey data for a damages model and to facilitate Notice to potential class members.
3. I produce my exhibit A, Conceptual Document, which reflect phase One.

1
2
3
4 I declare under penalty of perjury under the laws of the State of California that the
5 foregoing is true and correct.
6

7 Executed this 25th day of November 2016 at Los Angeles, California.

8
9
10 
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28

JOHN WELCHES
Declarant

Conceptual Documents

Author(s): Red Mallard Development Partner

Project Phase: Concept

Status: Final

Date: 11/04/16

Table of Contents

1	Introduction.....	3
2	Functional Description.....	3
3	Narratives / Application Flow	3
3.1	Mobile Application	3
3.1.1	Online Demo of Mockups	3
3.1.2	Login Flow.....	3
3.1.3	Participants.....	3
3.1.4	Questions Flow	4
3.1.5	Camera Flow	4
3.1.6	Fingerprint Flow.....	4
3.2	Admin Application.....	4
3.2.1	Online Demo of Mockups	4
3.2.2	Search Flow.....	4
3.2.3	Questions Flow	5
4	System Architecture	5
4.1	Client/Server Architecture	5
4.2	MULTI-Tenant	7
4.3	Client Support	7
4.4	Server Support	7
4.5	Cloud Architecture	8
4.6	Load Balancer.....	8
4.7	Web Server.....	9
4.8	Database Server	9
4.9	URL-Based Content Storage	9
4.10	Mail Server	9
4.11	Authorization Architecture	9
4.12	Device Registration	10
4.13	Account Registration.....	11
4.14	User Authorization	12

4.15	Deployment Architecture.....	12
4.16	Development.....	13
4.17	Staging.....	13
4.18	Production.....	13
5	Data Models	14
5.1	Application Data Models.....	14
5.1.1	Questions Table	14
5.1.2	Surveys Table.....	14
5.2	Smilebeacon Data Models	14
5.2.1	SFL_Users.....	15
5.2.2	SFL_Media	15
5.2.3	SFL_UserSettings	16
5.2.4	SFL_Roles	16
5.2.5	SFL_UserRoles.....	16
5.2.6	SFL_Devices	16
5.2.7	SFL_UserDevices.....	17
5.2.8	SFL_Domains.....	17
5.2.9	SFL_Globals.....	17
6	REST Services	19
6.1	Application REST Services	19
6.1.1	Application Services.....	19
6.1.2	Admin Services	20
6.2	Existing Smilebeacon REST Services	21
6.2.1	User Services.....	21
6.2.2	Device Services	23
6.2.3	Media Services.....	23
7	Appendix.....	25
7.1	Mobile Application.....	25
7.2	Admin Application.....	31

1 Introduction

Conceptualize a mobile application and backend services to collect information from groups of fisher people at multiple locations, as defined within the Court approved Class Definition.

2 Functional Description

This will be an Android application that will collect form data and store it in a secure cloud service. Participants will be asked to provide information in the form of a survey.

Images of the participant will be obtained using the front-facing camera of the mobile device provided contemporaneously with the completion of the survey.

Fingerprint images will be obtained contemporaneously using an external fingerprint peripheral that is connected to the device via USB.

The application will ask for an image capture, when prompted, the participant hold in shot, a form of official identification. Once the capture is complete, the application will convert the images to a JPEG format and then upload the image to a cloud-based storage service. After the completion, the URL generated will be saved with the survey in the database. This goes for facial image capture and fingerprint capture.

To ensure the authenticity of the process, the geolocation and the date of when the survey was filled out will be stored in the database. This will allow a third party to confirm the longitude and latitude of the participant at the time the survey was completed and if necessary cross reference identify against a contemporaneous photograph, government photo identification, fingerprint capture and geolocation showing GPS, time and date.

3 Narratives / Application Flow

The following section explains the intended user flow. These are to be read alongside with the attached mockups found in the Appendix.

3.1 Mobile Application

3.1.1 Online Demo of Mockups

This interactive tool demos the flow of the mobile application described in the Narratives.

- <https://mockingbot.com/app/vWsQXs0iE85k6FuCDiqC7CZ62uDM3xh>

3.1.2 Login Flow

Starting up the application, the screen will present a module to log in with email and password. Only authorized users will have credentials to log into the application. After they can proctor for the survey takers.

3.1.3 Participants

Subject to judicial approval, notice will be given in the local communities as defined by the scope of the class, calling upon all those who qualify and wish to participate, to attend, subject to certain limiting factors, and participate in the survey.

This screen is for survey takers to look up their existing ID if they were contacted prior to taking this survey. If the user does not have one, they can opt out and continue to the next screen. If they do have an ID, the ID would be submitted to see if it exists. If it was found, they would have to confirm the identity to continue.

- If ID is found:

- “ID found! Are you {{Name}}” - not sure if name could be spoken
 - “Yes” will continue forward to the Personal Info screen
 - “No, Try Again” will go back to the ID lookup screen
- If ID is not found:
 - An error would just pop: The submitted ID could not be found. To continue, select “No, I don’t have an ID yet”

3.1.4 Questions Flow

Instructions can be accompanied with audio narration in the preferred dialect. This flow shows how the questions would be displayed to the survey taker. Initially they would have to fill out the information form to collect some data about them.

- User would fill in their personal info
 - Do you have government approved photographic identification with you (required – if NO then unable to proceed further)
 - Christian Name (required)
 - Family Name (required)
 - Community (required)
 - Compound (required)
 - Mobile Phone Number
 - Email

The user would be instructed to take a picture of themselves, clearly holding their photographic identification, before proceeding to the next screen where they would then take the survey.

3.1.5 Camera Flow

The user will be instructed to take a picture of themselves right after filling out the form. They will also be instructed to take another one randomly during the survey questions to ensure continuity of participant.

3.1.6 Fingerprint Flow

Once the user is deemed eligible, they will be instructed to use the attached fingerprint device to scan their finger. After a successful read, the user will be informed and the process is complete. In the case of an unsuccessful read, a pop up will instruct the user to try again. After completion, the device can be returned to the proctor to set up a new form.

3.2 Admin Application

3.2.1 Online Demo of Mockups

This interactive tool demos the flow of the admin application described in the Narratives.

- <https://mockingbot.com/app/unNLz2LPyBHqKy3RJeyVwSyl9epJTvS>

3.2.2 Search Flow

The administrator web pages allow for admin users to look up and only read the data. It cannot be edited or changed. The surveys can be searched by ID, name of the survey taker, email, phone number, community, compound, and the date of the survey taken. Searching without any fields filled in (empty search) will return all the surveys. Clicking search will pull the data with the search fields parameters and will populate the table below.

The user can adjust the number of results per page by selecting the drop down to show from 25 to Display All. If the number of results are greater than the number of results per page, then paging will occur. Results will clearly show current results versus the total number of results.

The date will be displayed in a table with each row representing a survey taken. To view the survey's pictures, clicking in the row under the Portraits column will pop up the two portraits taken during the survey process. Same goes for viewing the fingerprint, clicking under the Fingerprint column will pop up the fingerprint taken during the survey process.

The list will be available to export to an Excel Sheet.

3.2.3 Questions Flow

An administrator user can create, edit, and delete questions. The table will display all the current questions in the database. The order will be determined by the user entered value for that column. This data is fed back to California wirelessly and in real time to allow the damages model.

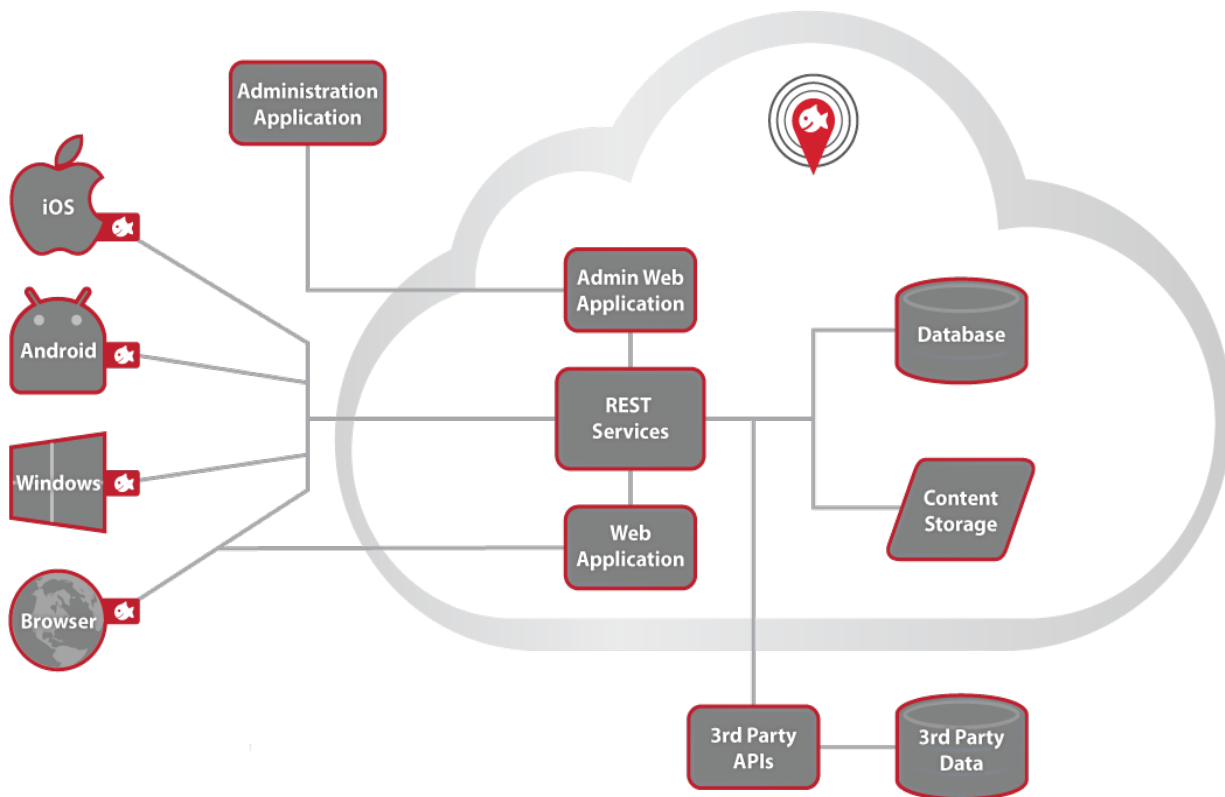
4 System Architecture

The application utilizes the Smilebeacon libraries and backend services. The following brief description of these components covers:

- Client/Server Technologies
- Identification of 3rd Party Libraries and Services
- Server Technologies & Scalability
- Security, Privacy, and IP Protection
- Supported Technologies (Client or server OS, Device Versions/Screen Sizes, etc.)
- Planning of Side-by-Side Development, Staging, and Production Environments
- User Authorization

4.1 Client/Server Architecture

Client/Server Architecture covers the interaction that client applications have with the database and web services. In future phases, the following may change as the system matures and grows with the IT infrastructure of tenants on the platform.



Client technologies include *iOS*, *Android*, *Windows 8*, and *Browsers*. All client services will retrieve and save data using the existing Smilebeacon APIs and custom APIs written for Tenants via REST Services.

Important architectural aspects of the server-side RESTful service API:

- Standard URL Endpoints
 - Each API method is exposed using a publicly addressable URL endpoint. This allows the API to be consumed by most any client application (desktop applications, web-based browsers, mobile phones, etc.).
- Autonomous and Stateless
 - Each exposed API method is constructed so that it can satisfy an entire atomic operation in a single call. This allows the REST Service API to SCALE-OUT by servicing client requests on multiple backend web servers.
- Secured using Authorization Tokens
 - Users that sign-in successfully are provided a security access token. This token is brokered in a stateless manner, ensuring that the method calls are secure and continue to maintain autonomy and statelessness. The token secures the method calls so that valid tokens with the correct scoping may grant access and serialization to certain data. If the token is not valid or does not have the appropriate scoping, then access and/or serialization of data will be denied.

As the Smilebeacon architecture provides for an administrative application ("the admin pages"), the Tenant's developers, administrators and support personnel are able to visualize, report on, and manage the Tenant's data directly.

4.2 MULTI-Tenant

Smilebeacon supports a MULTI-Tenant domain system. Each Tenant residing on Smilebeacon will be given an identifier (Domain Short Name) that uniquely identifies that tenant's data. This acts as a key that prevents data from other tenants being accessed by the web services or clients of other tenants in the platform.

The data between each Tenant will not be shared and will be fully secured.

During development, the application may be hosted as a tenant in Smilebeacon. For Production release, the application will be the sole tenant in its own installation of Smilebeacon with its own hosting account through Amazon Web Services (AWS).

At that point it would also be possible to host multiple applications as tenants within that installation.

4.3 Client Support

Currently, each client supports iOS versions 7 and higher, Android 4.0 and higher, Windows Phone 8.0 and higher, and varying browser versions (covered later). Support is included for both phone and tablet devices in specific orientations. The selected configurations allow development on current platforms and screen sizes. As we identify other device configurations that have a significant user base, support can be added at that time.

Note the following supported configurations:

- All Android devices capable of installing Android 4.0 or higher.

Support Matrix

Client	Any Screen Size	480x320px	Landscape	Portrait
Android Tables (Android OS 4.0+)	✓		✓	

4.4 Server Support

Tenants might require that a SMILEBEACON server environment be deployed into their IT infrastructure. The following is an example of a requirement specification:

Component	Version
	Microsoft Windows Server 2008 R2
Web Server	IIS 7.0
	ASP.NET MVC 4
Database	Microsoft SQL Server Enterprise 2008 R2
Content Storage	URL-Based File Storage

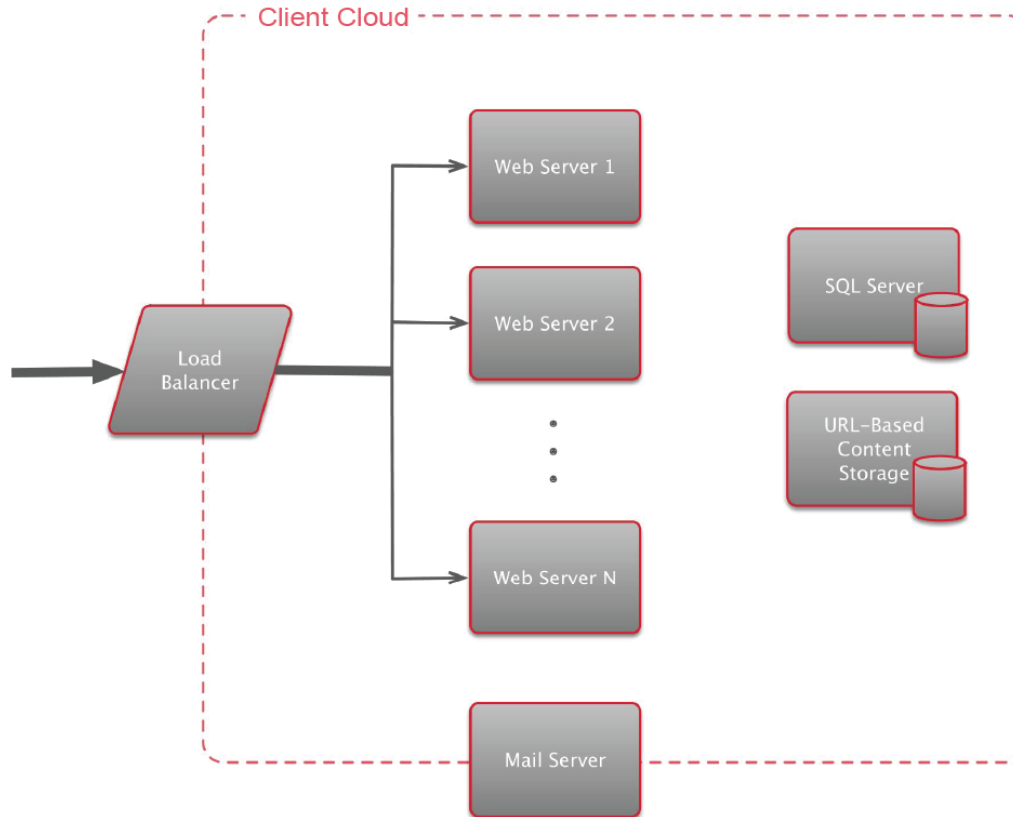
IIS Application Server on Windows hosts the REST API. IIS is a mature, highly available application hosting environment that can SCALE-UP to high powered hardware servers.

Microsoft SQL Server hosts the Tenant's data. Microsoft SQL Server is an enterprise level database server that is capable of storing high volumes of data while maintaining excellent performance. Microsoft SQL server can be replicated/mirrored in order to quickly and seamlessly handle disaster recovery scenarios and maintain high availability during server outages. As with the application hosting server, IIS, the Microsoft SQL server can SCALE-UP to high powered hardware servers.

Note that the infrastructure hosting the REST API and other server components may use the latest version of .NET, but will ensure backward compatibility for any platform components that the Tenant may integrate which require a lower version.

4.5 Cloud Architecture

Cloud architecture focuses on the ability of many components and systems to work together to deliver and process data for clients of the cloud. Cloud architecture has the ability to scale up, scale down, or to be reconfigured without interruption to clients. The architecture and the systems that drive the cloud architecture are fully transparent to the clients of the cloud. The composition of the components and systems that comprise the cloud should be positioned and configured in a way that maximizes performance, scalability, availability, recovery, and security. The following describes the general architecture of the Smilebeacon multi-tenant domain cloud environment.



4.6 Load Balancer

The Load Balancer is the named endpoint on the Internet. A single internet domain name will be directed to the load balancer. Although optional, a load balancer is necessary to scale out the web services and address high availability at the system level.

- Scale Out
 - As the volume of users and requests increases, more web servers can be added to the load balancer to handle the increasing volume.
- High Availability
 - The IIS hosting application on each web server addresses high availability at the software level by self-monitoring performance counters and restarting when those counters reach specified performance thresholds. The load balancer handles this at the system level by monitoring the health of the hosting application and specified system performance counters. If a web server becomes unavailable due to some unforeseen problem, then the load balancer will remove the

unhealthy web server from responding to incoming requests and allow the healthy servers to respond to the requests.

The load balancer is located in the public subnet. This means that the server can accept inbound and outbound connections from the Internet. The load balancer is only open to port 80 and port 443. Port 80 is open only to capture any incoming requests in order to redirect those requests back to the secure port 443.

4.7 Web Server

Web Server instances contain the REST API, Web Site, and Admin Pages. All web servers under the load balancer are exact copies of one another.

The web servers are located in the public subnet, but are firewalled so that incoming requests from any public IP address will not be accepted. This protects the web servers from outside attacks. The web servers will be able to send requests to public IP addresses. For instance, push notifications, emails, and SMS requests can be sent through to public IP addresses.

4.8 Database Server

The database servers are located in the public subnet, but is firewalled so that incoming requests from public IP addresses will not be accepted. Only the web servers within the private subnet will communicate with the database server.

4.9 URL-Based Content Storage

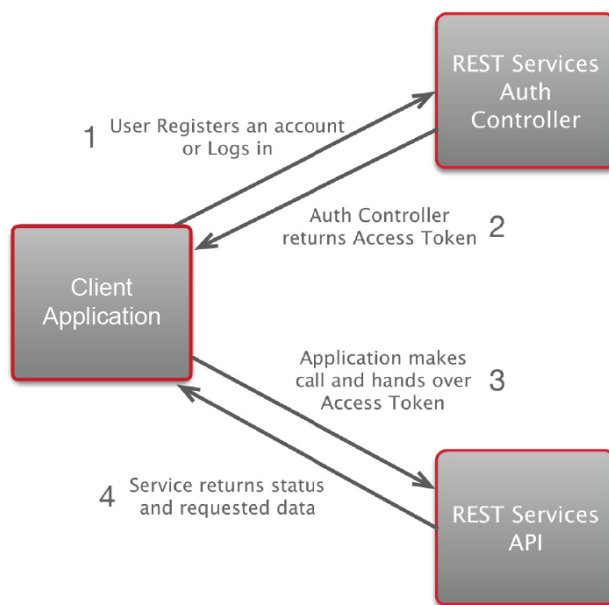
Content Storage being URL-Based is a very general requirement in which stored files are accessible via a public or private URL. These URLs must be unique for all content and will be stored in corresponding fields of the database. Similar to the SQL Server, these files are easily transferable to any other URL-Based Content Storage. Depending on the security requirements for the content, the URL may require additional query string parameters to specify authorization information.

4.10 Mail Server

The Mail Server is an outbound SMTP server that can be accessed by all server-side services in order to send email based on events or batch processes.

4.11 Authorization Architecture

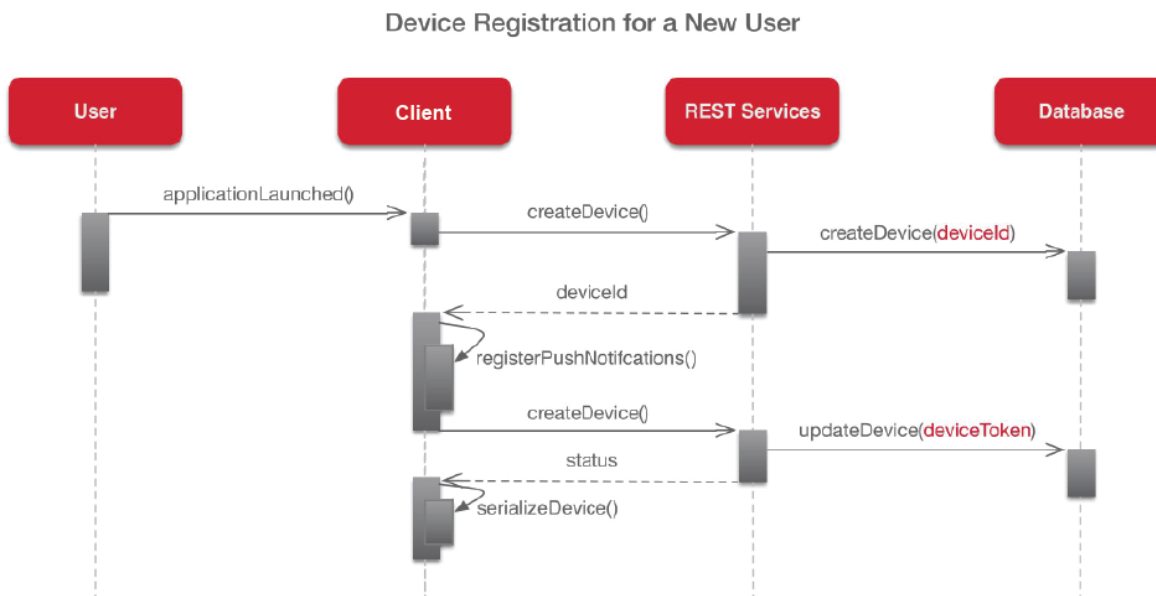
The following diagram depicts the authorization architecture.



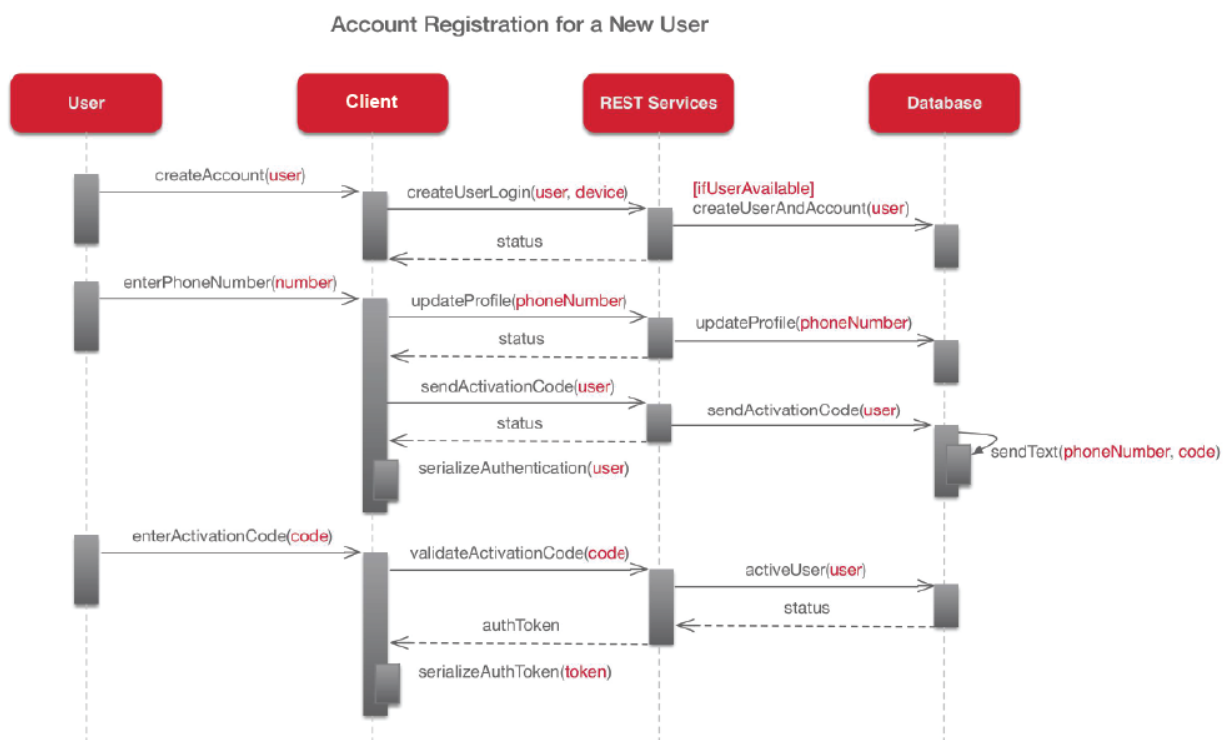
After Registering or Sign-In, the user receives an access token which validates their authenticity. With a valid access token, users can make REST API calls as long as permitted.

The following diagrams demonstrate the process of registering a device, registering an account, and subsequently authorizing the user.

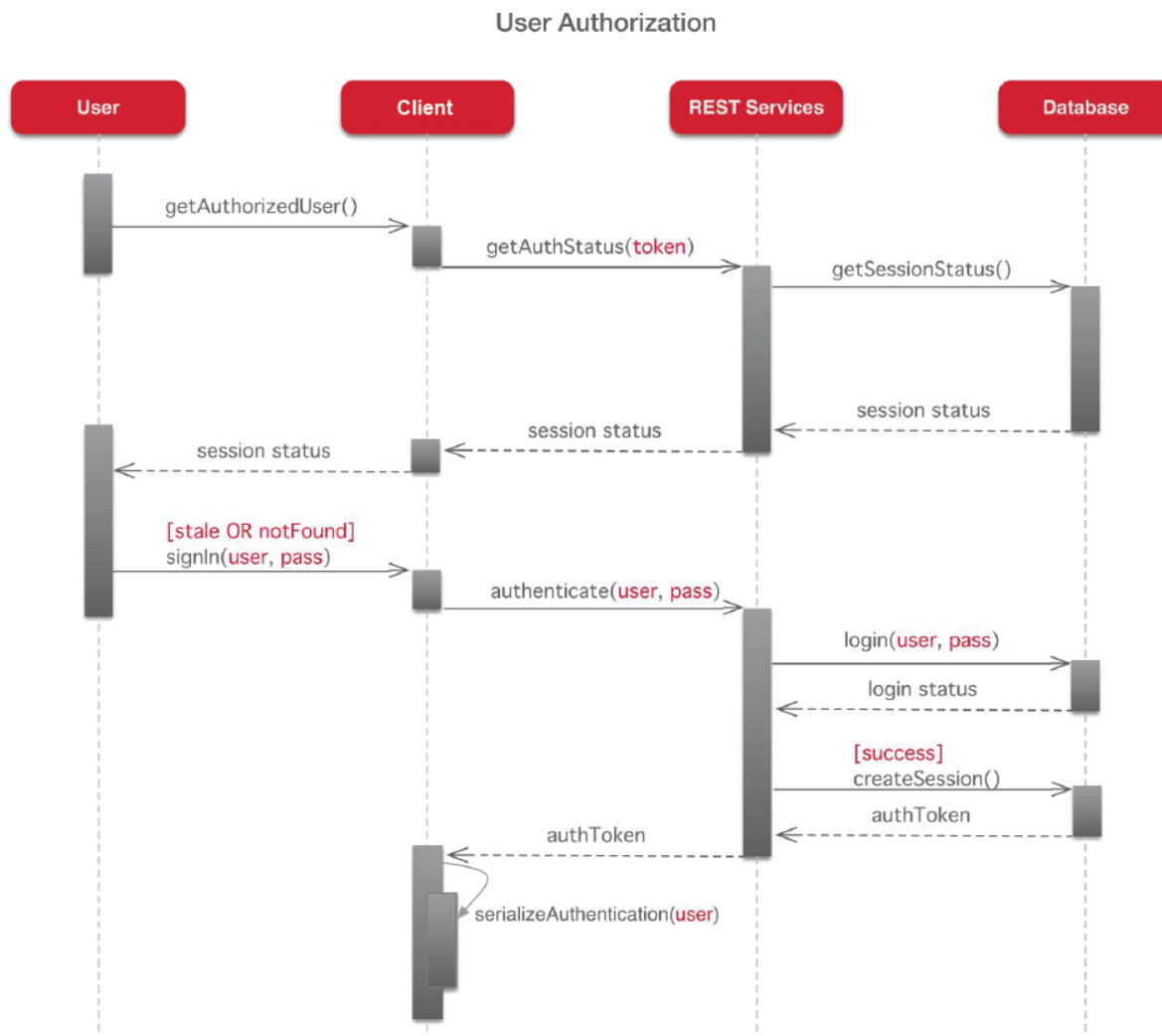
4.12 Device Registration



4.13 Account Registration

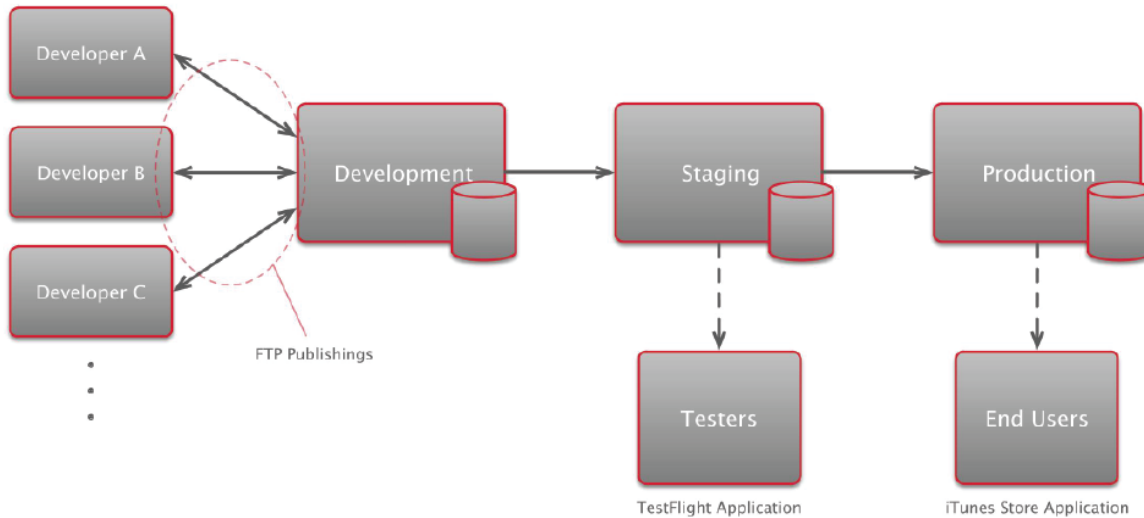


4.14 User Authorization



4.15 Deployment Architecture

Deployment Architecture covers the three-phase process that is typically used to deploy to web servers. Development (commonly just "Dev"), Staging, and Production make up these phases as well as their respective environments. The environments can be thought of as duplicated instances of the database and web services with their own data and code versions. The following diagram describes the deployment process.



4.16 Development

The Development Phase is when implementation of the application occurs, from start to finish. The environment is completely private to developers and shared amongst them all. Different developers are able to push and pull changes from the server, typically using version control to synchronize the Development environment's pristine version with the developer's local versions. Server code is published to the Development environment over File Transfer Protocol (FTP). In this stage, developers are responsible to unit test features. After a set of defined features have been adequately tested, the current deployment on dev is promoted to staging.

4.17 Staging

The Staging Phase is intended for thorough testing of the application by testers outside of the development group. In this environment, the server code is in a more stable state than that of Development. Therefore, this stage cannot be reached until there is a stable build able to be deployed for testing.

Clients pointing to the staging environment will be distributed through applications such as TestFlight, which makes the app available to certain users without having to officially release the application to the public. Testers can download and test the client application and effectively report any issues found.

When all defects in the staging environment have been addressed, the deployment is promoted to Production according to a well-defined rollout schedule.

4.18 Production

The Production Phase is when the application and web services are released to the public. In the grand scheme of things, this is the end goal for each implementation phase within software development.

This environment is then maintained continuously over the lifetime of the application.

5 Data Models

This section describes the structure of the data that is stored within the database.

5.1 Application Data Models

These are new models that are to be created for the Fisherman application.

5.1.1 Questions Table

Column Name	Type	Notes
QuestionId	Guid	Primary Key
Order	Int	Order of the questions
QuestionType	Int	Type of question (Y/N, drop down select)
Answers	JSON	List of possible answers

5.1.2 Surveys Table

Column Name	Type	Notes
SurveyId	Guid	Primary Key
Portraits	JSON	List of image URLs
ChristianName	String	First Name
FamilyName	String	Last Name
Email	String	
MobileNumber	String	
Community	String	
Compound	String	
Date	Date	
SurveyAnswers	JSON	Recorded answers from the survey
Fingerprint	String	URL of fingerprint image

5.2 Smilebeacon Data Models

Please note that each table in the Smilebeacon platform area of the database contains a DomainShortName. This serves as a key allowing Smilebeacon to maintain the values of each entity of all tenants within the following tables. This also prevents data from being shared between tenants even though the data is housed all within the same tables in the same database. For example, the users of Tenant A and Tenant B are both stored in the SFL_Users table. The Smilebeacon REST services inspect the DomainShortName property to ensure that Tenant A's application only has access to Users with the corresponding DomainShortName.

For Red Mallard, in production, the application will be hosted on its own installation of Smilebeacon but the application will still be a tenant in that installation. In the future, other the application applications could be added as tenants of Red Mallard's installation. At that time, it may be beneficial to customize this behavior so that users of any Red Mallard application can login to other Red Mallard applications using the same username/password combination.

5.2.1 SFL_Users

This table maintains the record of all User Accounts in the Smilebeacon system.

Column Name	Type	Notes
UserId	Guid	PrimaryKey
DomainShortName	String	Abbreviated name for Tenant Domain
UserName	String	Email address
Password	String	Encrypted Password
ActivationCode	String	Code for account activation
ActivationTimeStamp	DateTime	Account activation Date and Time
FirstName	String	
LastName	String	
RescueEmail	String	
Gender	Bit	
DOB	Date	Date of Birth
CellNumber	String	
ImageId	Guid	FK SFL_Media
StreetAddress	String	
StreetAddress2	String	
City	String	
State	String	
Zip	String	
Country	String	
About	String	Short description of the User (user-generated)
PasswordResetToken	Guid	Token for Resetting Password
CurrentDeviceId	Guid	DeviceId for user's most recently used device
Suspended	Bit	
CreateDate	DateTime	Creation Date
ModifiedDate	DateTime	Modification date
RequestPasswordDate	DateTime	Date when user requested a Password reset
PasswordExpiration	DateTime	Date for password expiration
FacebookLink	String	Link to user's Facebook account
LinkedinLink	String	Link to user's Linkedin account

5.2.2 SFL_Media

This table maintains references and metadata for storing and accessing user-generated Media files used in Smilebeacon System such as images, videos and sounds.

Column Name	Type	Notes
MediaId	Guid	PrimaryKey
MediaBucket	String	Bucket name for Amazon Simple Storage Service (S3)
MediaThumbnailBucket	String	Thumbnail Bucket name for Amazon Simple Storage Service (S3)
MediaStatus	Int	Different Status values for a specific Media Object
MediaType	Int	Type of Media (e.g. Image, Video, etc.)
MediaFileName	String	Name of the Media Object
MediaDescription	String	Description of the Media Object

DomainShortName	String	Abbreviated name for Tenant Domain
CreateDate	DateTime	Media Object Creation Date
ModifiedDate	DateTime	Media Object Modification Date
MimeType	String	
URL	String	Amazon Simple Storage Service URL for the Media Object

5.2.3 SFL_UserSettings

This table maintains application-specific preferences ("settings") for a given user account.

Column Name	Type	Notes
UserSettingId	Guid	PrimaryKey
UserId	Guid	FK SFL_Users.UserId
PropertyName	String	Name for individual Setting
PropertyValue	String	Value for individual Setting
DomainShortName	String	Abbreviated name for Tenant Domain

5.2.4 SFL_Roles

This table maintains the different Roles that exist in the Smilebeacon System (e.g. Administrator).

Column Name	Type	Notes
RoleId	Guid	PrimaryKey
Name	Guid	Name of the Role
DomainShortName	String	Abbreviated name for Tenant Domain

5.2.5 SFL_UserRoles

This table maintains the record of specific roles assigned to a given User.

Column Name	Type	Notes
UserId	Guid	FK SFL_Users.UserId
RoleId	Guid	FK SFL_Roles.RoleId
DomainShortName	String	Abbreviated name for Tenant Domain

5.2.6 SFL_Devices

This table maintains the record of all Devices registered in the Smilebeacon system.

Column Name	Type	Notes
DeviceId	Guid	PrimaryKey
DeviceType	String	Type of the Device (e.g. iPhone 6)
DeviceInfo	String	Description of the Device
CreateDate	DateTime	Device Creation Date
ModifiedDate	DateTime	Device Modification Date
DeviceToken	String	Token for Push Notification
BeaconUUID	Guid	(not used for the application)
BeaconMajor	Int	(not used for the application)
BeaconMinor	Int	(not used for the application)

DomainShortName	Bit	Abbreviated name for Tenant Domain
-----------------	-----	------------------------------------

5.2.7 SFL_UserDevices

This table maintains the record of Devices related to a given User.

Column Name	Type	Notes
UserId	Guid	PrimaryKey
DeviceId	Guid	FK SFL_Devices.DeviceId
DomainShortName	String	Abbreviated name for Tenant Domain
CreateDate	DateTime	Device-User Relation Creation Date
ModifiedDate	DateTime	Device-User Relation Modification Date

5.2.8 SFL_Domains

This table maintains the record of Tenant Domains in the Smilebeacon System.

Column Name	Type	Notes
DomainId	Guid	PrimaryKey
Name	String	Tenant name
ShortName	String	Description of the Device
APIKey	Guid	Device Creation Date
Description	String	Device Modification Date
Url	String	(not used for the application)
PushCertificateProd	String	Production Certificate for Push Notification
PushCertificateDev	String	Development Certificate for Push Notification
PushPasswordProd	String	Production Push Notification password
PushPasswordDev	String	Development Push Notification password
S3BucketDev	String	Development Amazon Simple Storage Service Bucket Name
S3BucketProd	String	Production Amazon Simple Storage Service Bucket Name
SecurityLevel	Int	User-based authentication or application-access based on API key
AWSAccessKeyProd	String	Production Amazon Web Services Access Key
AWSSecretKeyProd	String	Production Amazon Web Services Secret Key
AWSAccessKeyDev	String	Development Amazon Web Services Access Key
AWSSecretKeyDev	String	Development Amazon Web Services Secret Key
VerifiedDomainName	String	Domain Name Verification
GCMAPIKey	String	Google Cloud Messaging API Key
GCMSenderId	String	Google Cloud Messaging Sender Id

5.2.9 SFL_Globals

This table maintains Global settings used by the Smilebeacon System (e.g. Application Version Number for Forced Upgrade)

Column Name	Type	Notes
GlobalId	Guid	PrimaryKey
PropertyName	String	Name for individual Global Setting
PropertyValue	String	Value for individual Global Setting
DomainShortName	String	Abbreviated name for Tenant Domain

6 REST Services

The following details the new REST services that support the features of the application. It describes, in general terms, the input data types required of and output value types returned by the REST services. Types (data model entities) that are the domain of the database schema are described in the Data Model section. Some of the types described in this section are the domain of the client application – or more specifically the transportation layer between client and server – and their properties are not described in the Data Model section. For ease of reading, superscripted numbers denote these types and their definitions can be found in the endnotes of this document.

The verbs (HTTP Methods) of the REST services are described with the following general template:

VERB	Input Model	Output Model	Output Codes	Notes
VERB_NAME	Custom Data Type <i>Required Properties</i> AuthorizationCredential(s)	Custom Data Type Status	Possible Status Codes	Notes

6.1 Application REST Services

The following details the new REST services that are unique to the features of the application and must be created.

6.1.1 Application Services

<https://host/restservice/api/SurveyRest>

VERB	Input Model	Output Model	Output Codes	Notes
GET_QUESTIONS	AuthToken	ListOfQuestions Status	OK UserNotAuthenticated SessionExpired PasswordExpired ConnectionFailed Error	Each question will have a set of possible answers and answer type so the UI can account for the different types of answers.
SAVE_SURVEY	Survey AuthToken	Status	OK UserNotAuthenticated SessionExpired PasswordExpired ConnectionFailed Error	Saves the images, information, and answers that pertained to the session.
LOOKUP_ID	Id AuthToken	IdInfo (Must figure out what data we can get back from the lookup) Status	OK UserNotAuthenticated SessionExpired PasswordExpired ConnectionFailed Error	Looks up the ID the existing database of victims to merge data.

6.1.2 Admin Services

<https://host/restservice/api/SurveyRest>

VERB	Input Model	Output Model	Output Codes	Notes
GET_SURVEYS	Survey AuthToken	ListOfSurveys Status	OK UserNotAuthenticated SessionExpired PasswordExpired ConnectionFailed Error	Searches all the surveys that are saved into the database. Must be admin to execute
GET_QUESTIONS	AuthToken	ListOfQuestions Status	OK UserNotAuthenticated SessionExpired PasswordExpired ConnectionFailed Error	Same from application services
SAVE_QUESTION	Question AuthToken	Status	OK UserNotAuthenticated SessionExpired PasswordExpired ConnectionFailed Error	Editing a question. Must be admin to execute
CREATE_QUESTION	Question AuthToken	Status	OK UserNotAuthenticated SessionExpired PasswordExpired ConnectionFailed Error	Creating a question. Must be admin to execute
DELETE_QUESTION	Question AuthToken	Status	OK UserNotAuthenticated SessionExpired PasswordExpired ConnectionFailed Error	Deleting a question. Must be admin to execute

6.2 Existing Smilebeacon REST Services

The following details the available, existing REST Services of Smilebeacon and customization/enhancements needed for the application. Note that the application platform may not utilize all features available in Smilebeacon in the current project phase but they are described here regardless.

6.2.1 User Services

<https://host/restservice/api/UserRest>

VERB	Input Model	Output Model	Output Codes	Notes
AUTHENTICATE_USER	User Username Password DomainShortName APIKey	User Status AuthToken AuthTokenLT	OK UserNotAuthenticated UserNotActivated FailAPIKey PasswordExpired ConnectionFailed Error	If the username and password are valid, then the response will return a valid status and both a short and long term authorization tokens. The term for both long and short term authorization tokens are configurable in the database SFL_spGetIdleTimeout stored procedure.
AUTHORIZE_USER	User AuthToken	Status	OK UserNotAuthenticated SessionExpired ConnectionFailed Error	
REQUEST_PASSWORD_RESET	User Username DomainShortName APIKey	User Status	OK FailAPIKey ConnectionFailed Error	If the input Username, APIKey, and DomainShortName are valid, the server will generate and associate a reset token with the user. An email, SMS Text, or both will be sent to the user indicating how to reset the password and provides the reset token required to reset the password.
RESET_USER_PASSWORD	User Username PasswordRestToken NewPassword DomainShortName APIKey	User Status AuthToken AuthTokenLT	OK FailAPIKey ConnectionFailed Error	If the reset is valid, then the response will return information as if the client requested to authenticate. Or, in other words, the client will be signed on at this point.
CREATE_USER_LOGIN	User	User	OK	Creates a user

	Username Password DomainShortName APIKey *all other User model fields are optional	Status	FailAPIKey ConnectionFailed Error	account to maintain user profile information and to maintain authorization privileges. Although this verb creates the user account, the account is inactive and unusable until the 'activation step' is completed.
SEND_ACTIVATION_CODE	User Username Password DomainShortName APIKey CellNumber	User Status	OK FailAPIKey ConnectionFailed Error	Sends an activation code through SMS text messaging. An activation code is required to complete the final step in the 'create user login' flow.
ACTIVATE_USER	User Username Password DeviceId DomainShortName APIKey ActivationCode	User Status AuthToken AuthTokenLT	OK FailAPIKey ConnectionFailed Error	<p>A user is activated when the user provides the correct authentication parameters along with the correct activation code (that was sent through SMS text messaging).</p> <p>If successful, the user will be activated and will be immediately signed on. The AuthToken and AuthTokenLT will be valid and can be used to make authorized calls to the backend server.</p>
UPDATE_USER_SETTINGS	ListOfUserSettings UserId PropertyName PropertyValue AuthToken	ListOfUserSettings Status	OK UserNotAuthenticated SessionExpired ConnectionFailed Error	Takes a list of user settings and saves them. If an entry does not exist for a specified property value, then the entry is created.
GET_USER_SETTINGS	User UserId AuthToken	ListOfUserSettings Status	OK UserNotAuthenticated SessionExpired ConnectionFailed Error	Accepts a user and return a list of user settings associated with that user.

USERPROFILE_BY_USERID	User UserId AuthToken	User Status *All non-system fields will be populated	OK UserNotAuthenticated SessionExpired ConnectionFailed Error	Accepts a user id and returns the user profile information associated with that user.
UPDATE_USER	User UserId AuthToken *Only non-system fields with non-null values will be accepted for update.	User Status	OK UserNotAuthenticated SessionExpired ConnectionFailed Error	Updates properties on the specified user profile. A number of system related properties are not available for update. This includes the user logon name. Only non-null values supplied on input will be updated.
GET_USER_ROLES	User UserId AuthToken	ListOfUserRoles Status	OK UserNotAuthenticated SessionExpired ConnectionFailed Error	Returns a list of roles for the specified user.

6.2.2 Device Services

<https://host/restservice/api/DeviceRest>

VERB	Input Model	Output Model	Output Codes	Notes
CREATE_DEVICE	Device DeviceType DeviceInfo DeviceToken (optional and may be null) DomainShortName APIKey	Device Status DeviceId	OK FailAPIKey PasswordExpired ConnectionFailed Error	Registers the device on the backend server. A non-null device identifier (DeviceId) will be returned upon successfully registering the device.
UPDATE_DEVICE	Device DeviceId DeviceType DeviceInfo DeviceToken (may be non-null) DomainShortName APIKey	Device Status	OK FailAPIKey PasswordExpired ConnectionFailed Error	The DeviceId is used to key the update. This call is made available primarily to make updates to the DeviceToken - which may be remove or added by the client-user at any time.

6.2.3 Media Services

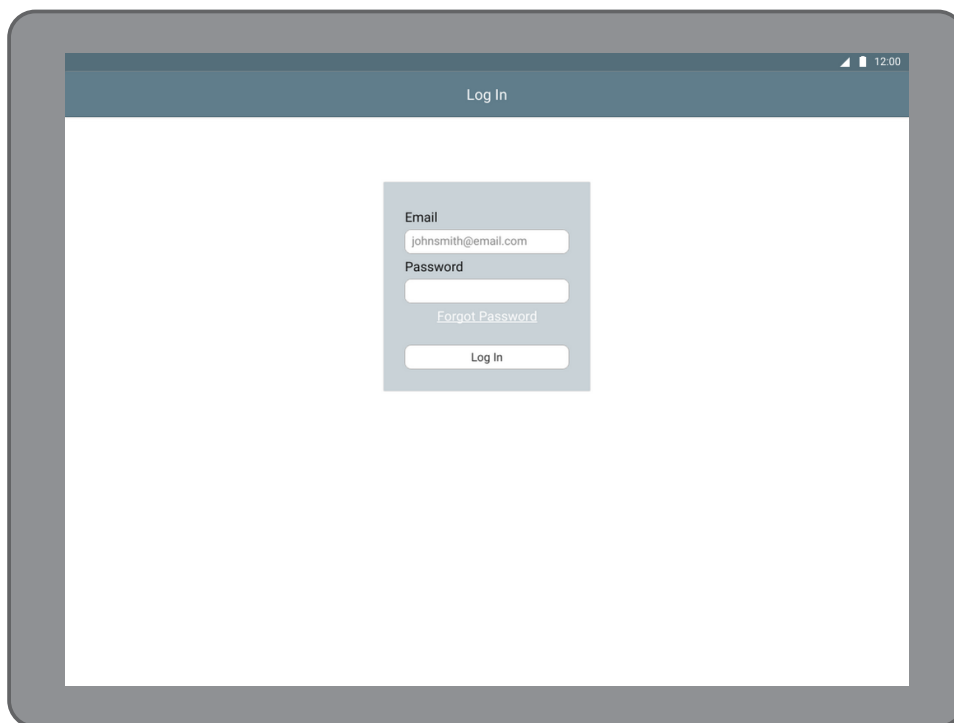
<https://host/restservice/api/MediaRest>

VERB	Input Model	Output Model	Output Codes	Notes
------	-------------	--------------	--------------	-------

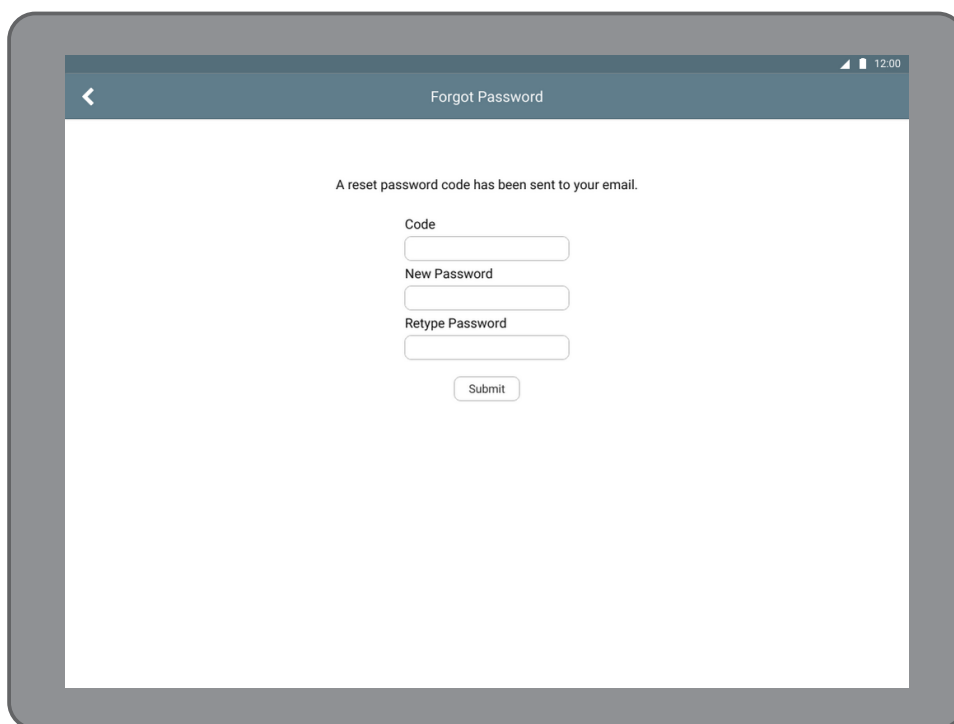
GET_MEDIA_GUID	Media Bucket AuthToken	Skin Status MediaId	OK UserNotAuthenticated SessionExpired PasswordExpired ConnectionFailed Error	Input an S3 bucket and return a media identifier.
UPDATE_MEDIA	Media MediaId *any non-null media field will be updated	Media Status	OK UserNotAuthenticated SessionExpired PasswordExpired ConnectionFailed Error	Updates any non-null Media field. A primary use case is to mark the media status once an asynchronous image file has been uploaded to a file server successfully.
SEARCH_MEDIA	MediaSearch *any non-null Media field will be searched	ListOfMedia Status	OK UserNotAuthenticated SessionExpired PasswordExpired ConnectionFailed Error	Search media by various fields.

7 Appendix

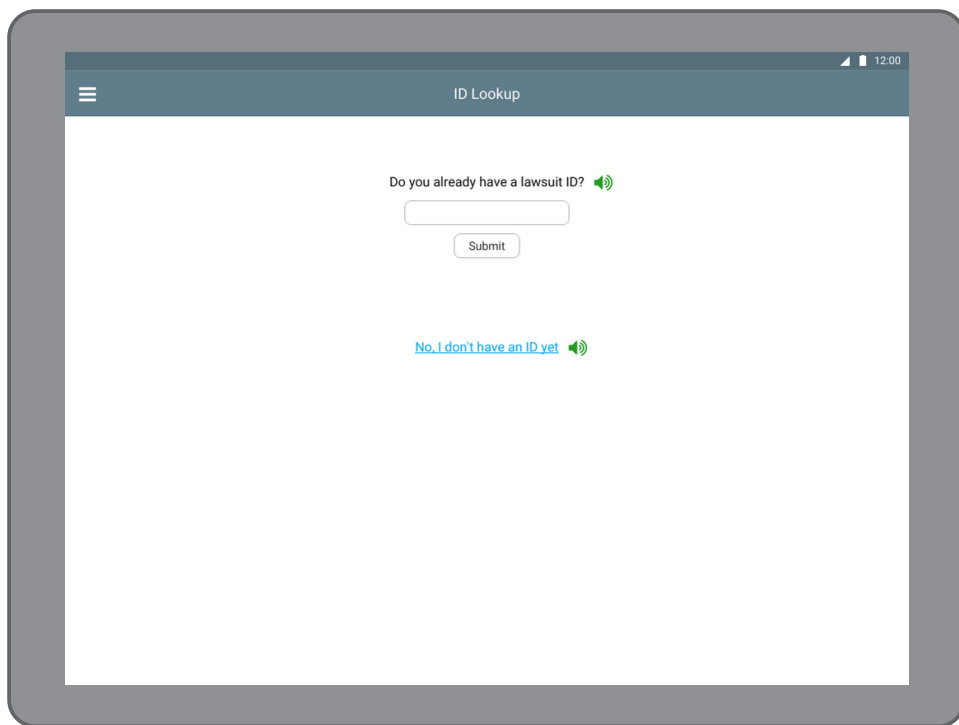
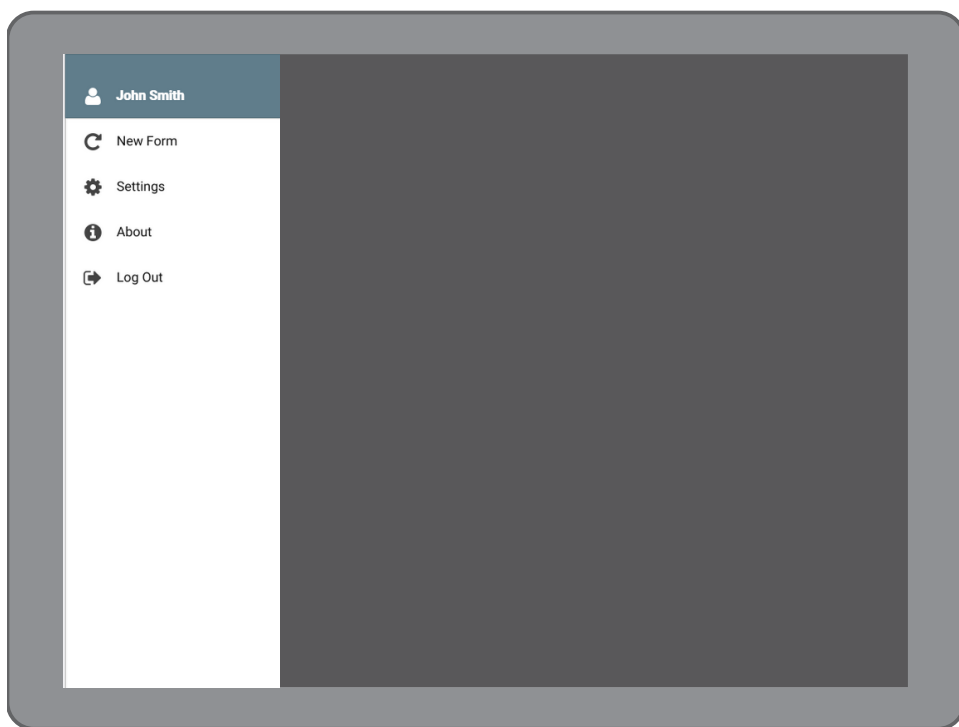
7.1 Mobile Application

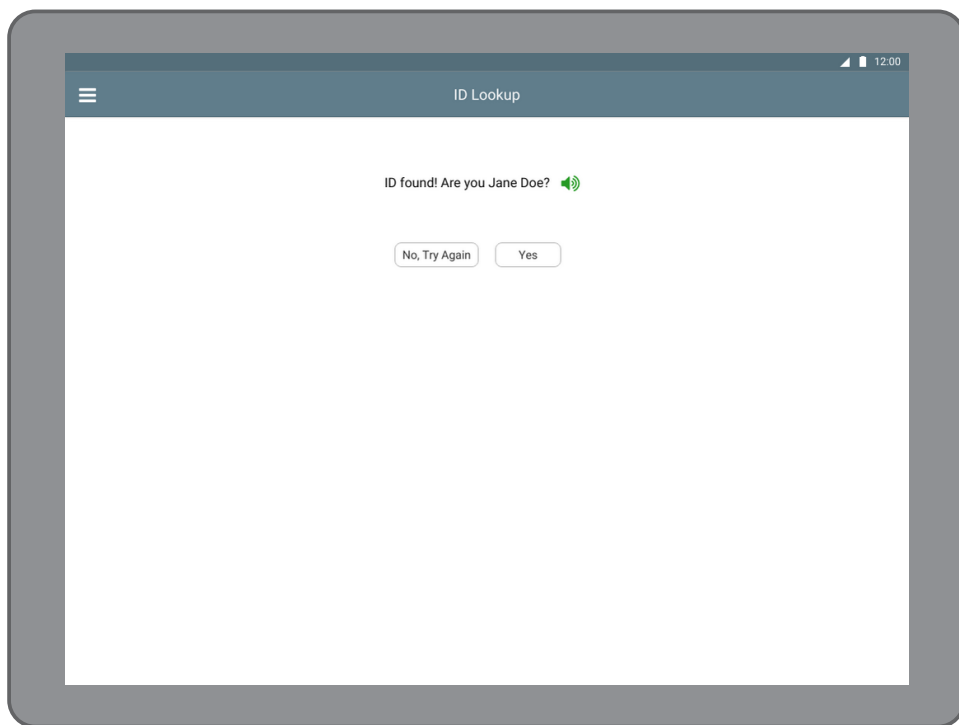


A screenshot of a mobile application's "Log In" screen. The screen has a dark blue header bar with the text "Log In" in white. Below the header, there is a light gray rectangular box containing the login form. The form includes an "Email" label, a text input field with the placeholder "johnsmith@email.com", a "Password" label, a text input field, a "Forgot Password" link, and a "Log In" button. The status bar at the top right shows a signal strength icon, a battery icon, and the time "12:00".




A screenshot of a mobile application's "Forgot Password" screen. The screen has a dark blue header bar with a back arrow icon on the left and the text "Forgot Password" in white. Below the header, there is a message: "A reset password code has been sent to your email." followed by three text input fields labeled "Code", "New Password", and "Retype Password". A "Submit" button is located below the input fields. The status bar at the top right shows a signal strength icon, a battery icon, and the time "12:00".



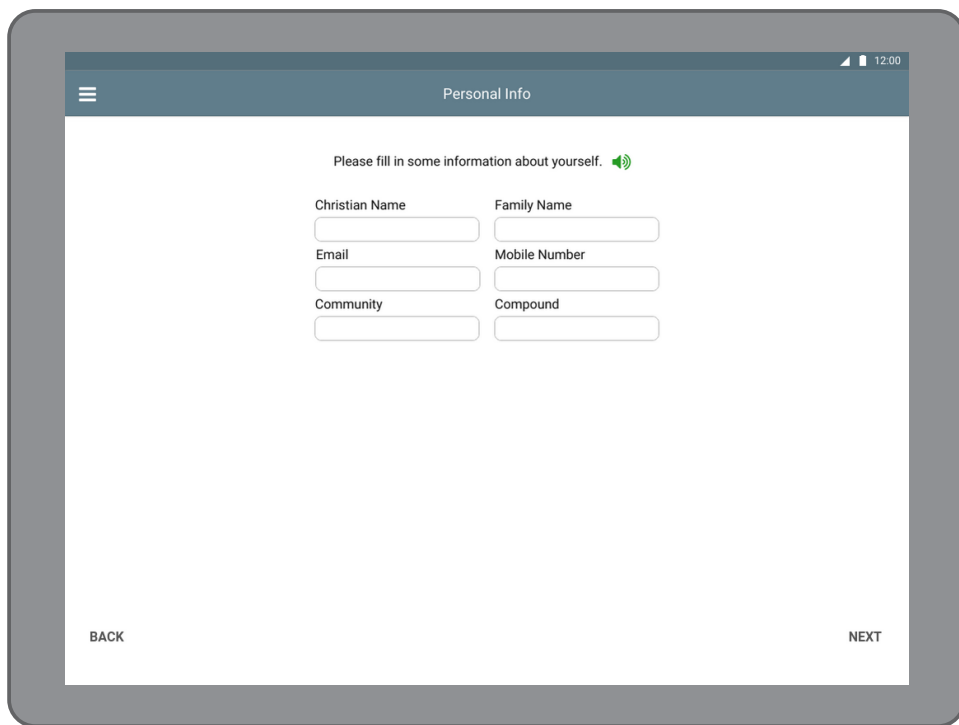


The image shows a tablet screen displaying an "ID Lookup" application. The top status bar shows the time as 12:00. The app's header is a dark blue bar with a hamburger menu icon on the left and the title "ID Lookup" in the center. The main content area is white and contains the text "ID found! Are you Jane Doe?" followed by a green speaker icon. Below this text are two buttons: "No, Try Again" and "Yes".

ID Lookup


ID found! Are you Jane Doe? 

No, Try Again Yes



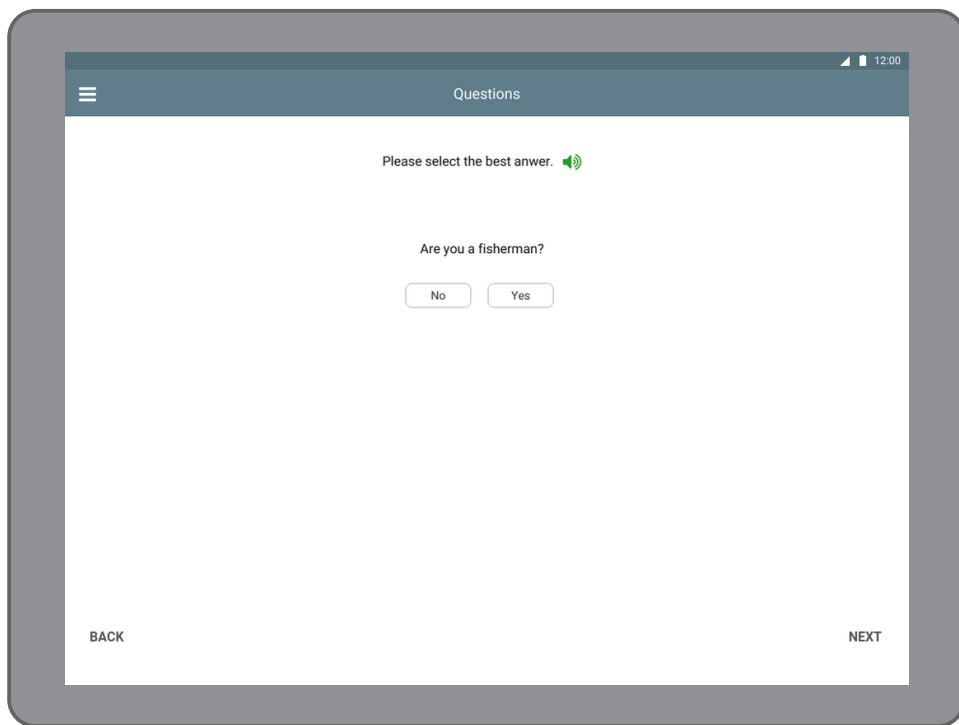
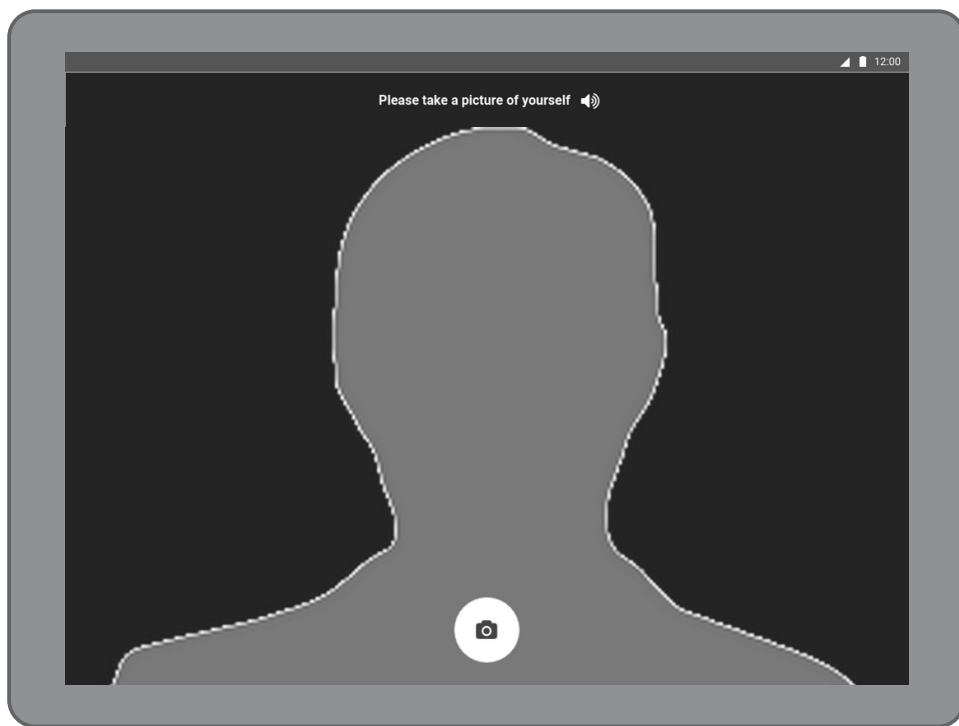
The image shows a tablet screen displaying a "Personal Info" application. The top status bar shows the time as 12:00. The app's header is a dark blue bar with a hamburger menu icon on the left and the title "Personal Info" in the center. The main content area is white and contains the text "Please fill in some information about yourself." followed by a green speaker icon. Below this text are six input fields arranged in two columns. The left column has fields for "Christian Name", "Email", and "Community". The right column has fields for "Family Name", "Mobile Number", and "Compound". At the bottom of the screen, there are two buttons: "BACK" on the left and "NEXT" on the right.

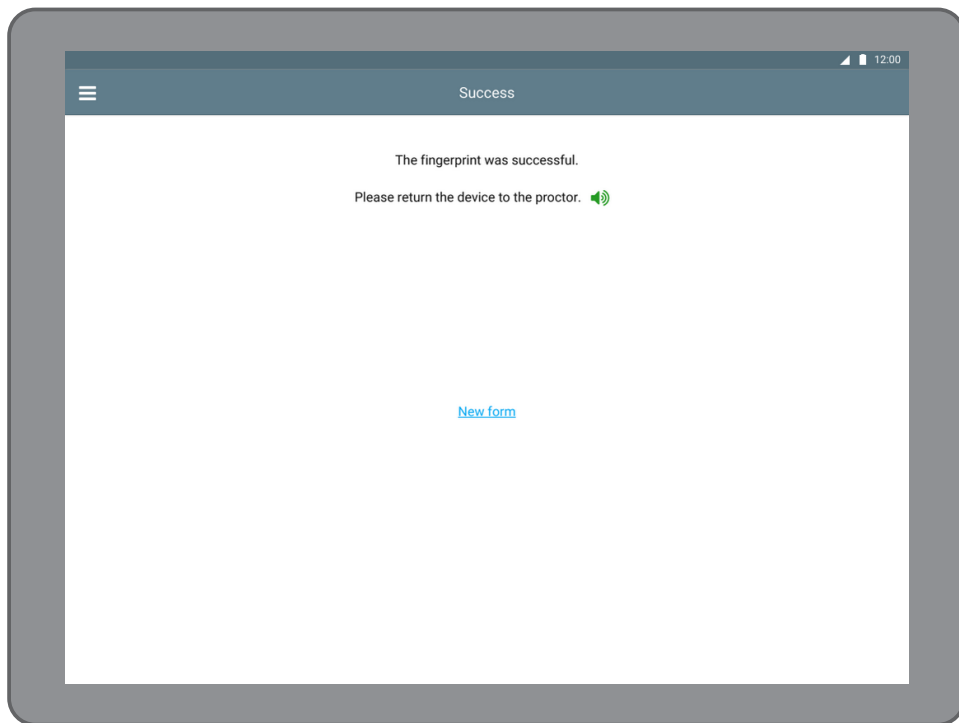
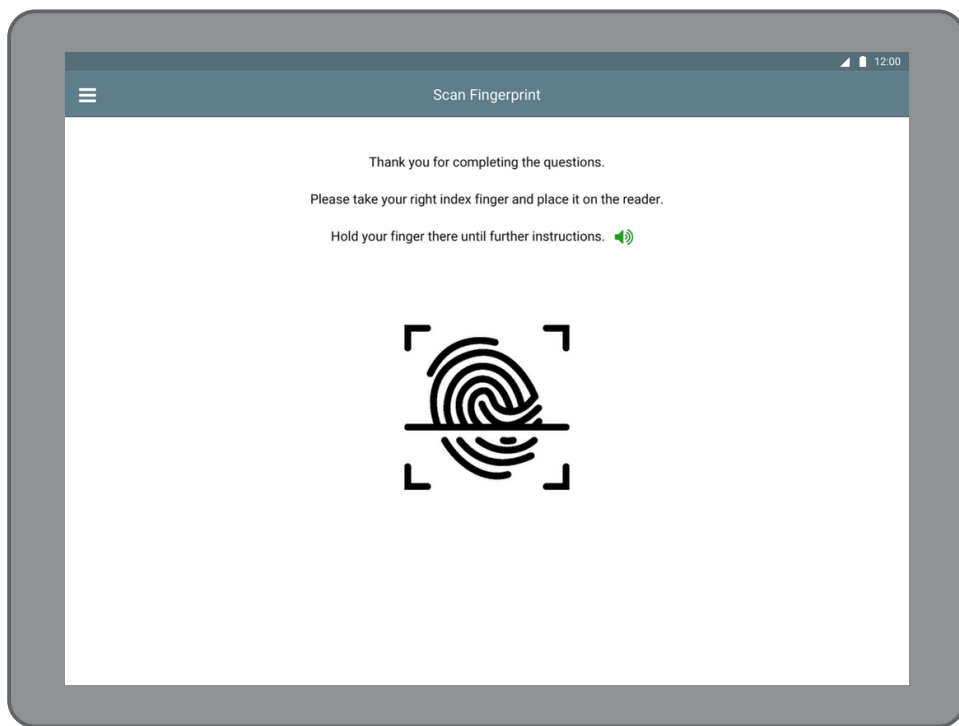
Personal Info

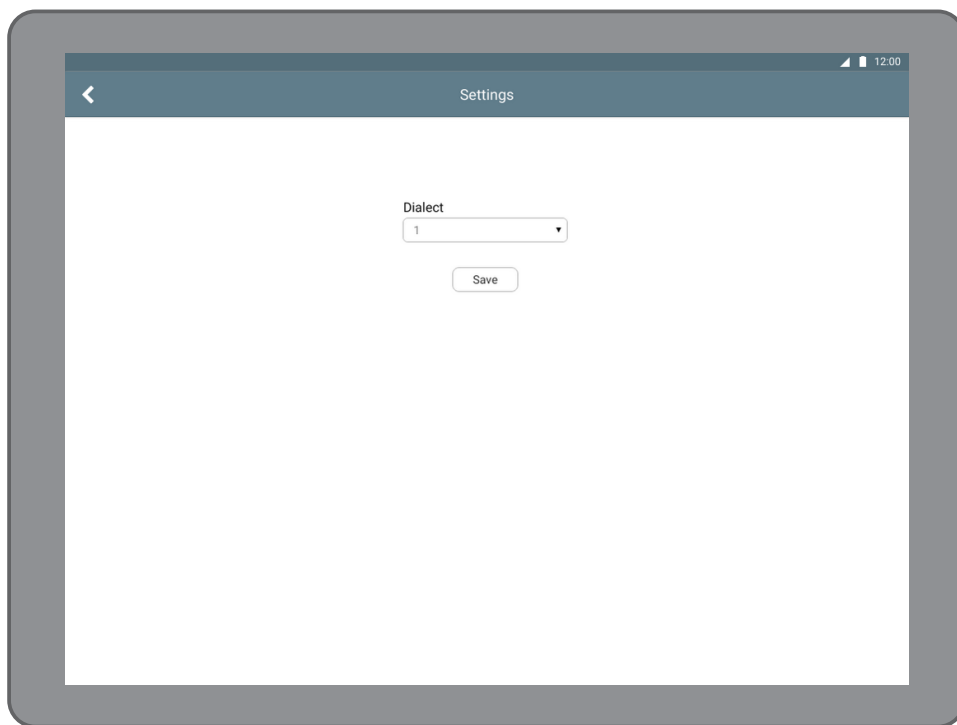
Please fill in some information about yourself. 

Christian Name	Family Name
<input type="text"/>	<input type="text"/>
Email	Mobile Number
<input type="text"/>	<input type="text"/>
Community	Compound
<input type="text"/>	<input type="text"/>

BACK NEXT







7.2 Admin Application

Surveys

SurveyID
Text Input

ChristianName
Text Input

Email
Text Input

Community
Text Input

StartDate
Date Input

FamilyName
Text Input

MobileNumber
Text Input

Compound
Text Input

EndDate
Date Input

Search

Showing Results: 1-25 of 100

Results per page: 25 [1](#) [2](#) [3](#) >

SurveyID	Portraits	ChristianName	FamilyName	Email	MobileNumber	Community	Compound	Date	SurveyAnswers	Fingerprint